CS112 - Fall 2022

Lab09

Instructor: Paul Haskell

INTRODUCTION

In this lab, you will develop some Java programs that read user input from the keyboard. A big focus of your work should be error-checking and -handling of incorrect inputs. You should assume (and the testers certainly will try) that user input will be incorrect. You will use the Scanner class to read the user inputs.

How do you want to test your programs? Type in entries for every test? Make and "redirect" text files with test cases? Build some Strings with test cases right into your program and test with those? A combination?

A simple start

Create a program called **EightLines.java**. This program should read in exactly eight lines of text (as returned by `Scanner.nextLine()`) from the keyboard input. Each line may contain 0 or more words. (A "word" is any set of non-space characters, separated from other words on the same line by one or more spaces.) Each input word should be printed alone on a line. Pretty simple!

What kinds of test cases can you think of?

- Lines with 0, 1, or more than 1 words
- Fewer than 8, exactly 8, or more than 8 input lines

If an input line contains no words, what should the program do? The specification says only to print words at the input, so such lines should be "counted as lines" but should not yield any output.

What should the program do if there are fewer than 8 lines of input? The program specification has not said yet and you are very entitled to ask. In this case, the program should use `System.err` to print the message "ERROR: fewer than 8 input lines" .

Reading doubles

Next, create a program called **EightDoubles.java** that reads 8 double values from the keyboard input and prints:

- Minimum <<whatever the right answer is>>
- Maximum <<whatever the right answer is>>
- Average <<Average value, rounded to 2 decimal places>>

In about a week we will learn about built-in functions to perform rounding. For now, you can use this technique:

```
double valueToBeRounded = ???;

double roundedValue = ((int) (100*valueToBeRounded))*.01;
```

For error handling, if the user enters an incorrect (non-numeric) input, you should prompt the user (repeatedly if needed) to reenter it. If the input contains fewer than 8 doubles, use `System.err` to print a useful message that contains the word "ERROR".

Quadratic Formula

Please write a program called **Quadratic.java**. Remember the quadratic formula? If you have the equation

$ax^2 + bx + c = 0$

The solutions are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

There are two solutions, one where the ± is a "+" and one where we take the "-".

Your program should prompt the user to enter values for a, b, and c and should read their values from the keyboard input. The program should print

x1 = <<first solution>>

x2 = <<second solution>>

You can use the built-in function `Math.sqrt()` to calculate the square root.

But if the solution is a complex number (i.e. if the value inside the square root is < 0) or is "degenerate" (a = 0), then do not print out any x values. Instead, print an error message using `System.err`. Include the word "ERROR" in your error message.

Fractions

Ok, now for a trickier program. Write a program called **Fractions.java**. This program will read in text specifying a fractional number and will print out the decimal value. For instance, if the input is "1/4" then the output should be 0.25.

The first step is to think about the exact format of the input. Our input might start with a '+', a '-', or just a number. After the number, there may be a slash ('/') or may be one or more spaces. If spaces, there may be following another number, which must be followed by a slash and then another number, for example:

-2 3/4

To keep things simple, let's not allow any decimal points in our input. And let's read the input as a single command-line argument rather than as keyboard input. Do ignore spaces at the start and/or end of the input.

If the input contains anything other than the given pattern, print an error message using `System.err`. Otherwise, print the decimal value on `System.out`.

Reminder

Put useful comments into your code. Design and organize your code so that it is easy for others to understand. Use reasonably short methods, and a mix of public and private methods as needed.

Put all your files in **Lab09** and push to GitHub before the deadline. This assignment must be turned in before Friday Sept 30th at 11:59pm.

Conclusion

This exercise gave hands-on experience coding with Scanner, but more usefully it pushed you to think about, design, and test for robust input handling. As you develop and improve your software for user input handling, you will find yourself reusing it frequently.

Grading Rubric

The first three programs are worth 10 points each: 2 points for each of 5 test cases.

The Fractions.java program is worth 30 points:

- 4 points for each of 5 tests cases correctly handled
- 0-10 points for the graders' evaluation of your software clarity and design quality